(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 June 2000 (29.06.2000)

PCT

(10) International Publication Number
WO 00/38078 A1

(51) International Patent Classification[6]: G06F 15/16, 15/173, 17/60

(21) International Application Number: PCT/US99/30580

(22) International Filing Date:
21 December 1999 (21.12.1999)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/113,094 21 December 1998 (21.12.1998) US

(71) Applicant: JJ MOUNTAIN, INC. [US/US]; 2013 Landings Drive, Mountain View, CA 94041 (US).

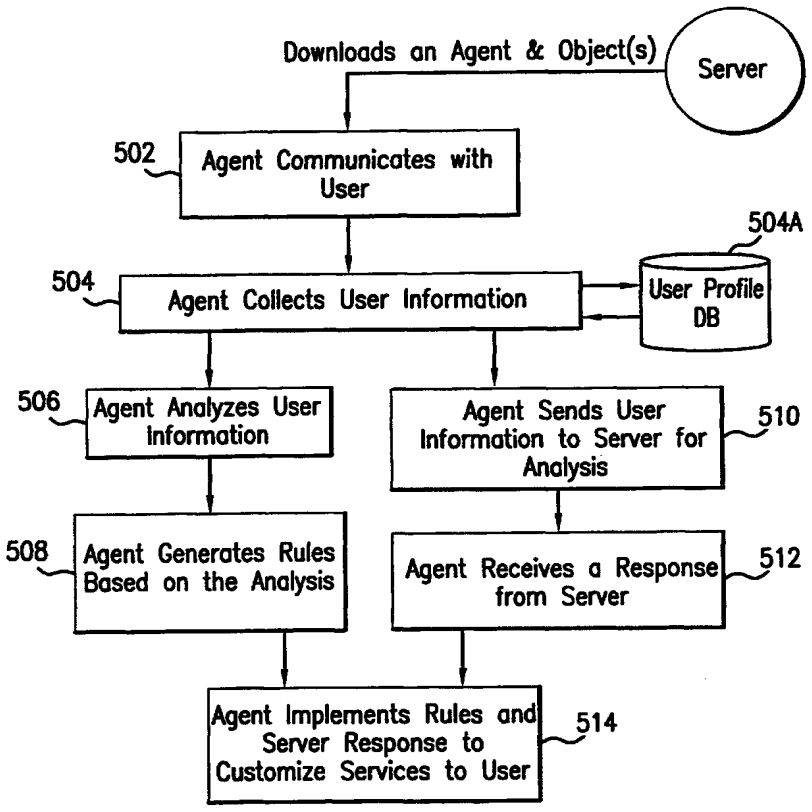
(72) Inventors: CAO, Jingjun; Apartment 6, 227 Pettis Avenue, Mountain View, CA 94043 (US). CHU, Chien-Yi; Apartment 6, 227 Pettis Avenue, Mountain View, CA 94043 (US).

(74) Agents: FRIEBEL, Thomas, E. et al.; Pennie & Edmonds L.L.P., 1155 Avenue of the Americas, New York, NY 10036 (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: METHODS AND SYSTEMS FOR PROVIDING PERSONALIZED SERVICES TO USERS IN A NETWORK ENVIRONMENT



(57) Abstract: Methods, systems and software products for providing personalized services to users in a computer network environment comprise collecting user information through at least one user interface (504), analyzing the collected user information (506), generating at least one rule based on the analysis (508), and providing at least one personalized service to the user based on the generated at least one rule (514). In another exemplary embodiment, the methods, systems and software products also comprise sending the collected user information to a server for analysis (504 and 510), receiving a response from the server based on the server analysis (512), and processing the received response (514). In one embodiment, processing comprises (i) saving the response in a local cache, and (ii) implementing the response.

WO 00/38078 A1

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

(48) **Date of publication of this corrected version:**
10 May 2001

(15) **Information about Correction:**
see PCT Gazette No. 19/2001 of 10 May 2001, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHODS AND SYSTEMS FOR PROVIDING PERSONALIZED SERVICES TO USERS IN A NETWORK ENVIRONMENT

This application claims priority of the provisional patent application entitled "A Distributed Intelligent Agent System", Serial Number 60/113,094, filed December 21, 1998, which is incorporated herein by reference for all purposes.

Field of the Invention

This invention relates to methods and systems for providing personalized services to users in a network environment.

Background of the Invention

In a network, multiple computers, including server computers ("servers") and client computers ("clients"), are connected to each other. Servers on a network typically provide services which are requested by clients. Services provided by servers include transferring useful information (i.e., reference material), entertainment, downloadable software programs, and more. In a wide area network, such as the Internet, users at client computers often need to spend a large amount of time searching for desired services. As the amount of information available on a network

such as the Internet continues to increase, the users's burden to efficiently use the network also increases. Thus, it is desirable to provide users with personalized services and assistance to navigate and use information in the network.

Some existing systems on the Internet attempt to provide personalized services. For example, there are web sites on the Internet that allow a user to create a personal profile. In this type of systems, users are typically prompted to fill out a form. The form usually allows the user to specify personal information including preferences. Any entered information is stored either locally or remotely. Typically, each time a user wants to access his personal profile, he needs to log-in to the server by typing in his username and/or password. Once within his profile, the user could obtain personalized services. Examples of this type of systems include "My Yahoo" at www.Yahoo.com, and "My Excite" at www.Excite.com.

There are many disadvantages in this type of systems. For example, user information is not automatically collected and there is virtually no update of the information unless the user initiates by editing his profile. Another disadvantage is that the user information is generally only usable at the web site where the user filled out the form. In addition, a form can only contain a limited number of questions and space for answers; thus, some personal information is necessarily not collectible.

Another type of existing systems provides personalized information to users after the users answer a series of questions. Answers provided by the users are used to prioritize and select the presentation of available information, which are maintained in server databases. An example of this type of systems can be found at: www.personalogic.com. In this system, a user first initiates the process by choosing an interested category. Next, the user is prompted to rank the importance of various characteristics in the interested category. Typically, the user marks boxes identifying each characteristic as "less important," "no opinion," or "more important." Based on the user's inputs, this system outputs a list of items in the interested category which should meet the user's overall preferences.

One problem with this system is that the user has to actively initiate and participate in the preference deduction process, which can be a time consuming task. Another problem with this system is that information entered into one web site is not usable at another web site. In addition, this system is static, namely, the answer to one question does not affect the presentation of other questions.

Yet another existing system attempts to personalize selections offered to a user by monitoring and compiling user information, such as the user's buying history or browsing habits. Using compiled information, this system can generate a list of recommended items. In some implementations, this type of systems can also take into account statistics (i.e., age, gender, residence location, employment, etc.) gathered among different users. For example, this system may use demographic characteristics to deduce the type of items that would be of interest to a particular user. An example of this type of systems can be found at: www.amazon.com. One major drawback of this type of systems is that user information can only be used at the web site that monitors and compiles the information. In addition, although compilation of information is continuous while the user is visiting the web site, typically, compilation activities cease when the user leaves the web site.

Other existing systems attempt to provide more user-friendly services. For example, some proprietary software developers have included animated figures in their software packages to provide a more user friendly technical support environment. For instance, instead of using a traditional window-styled help menu, some software products include an animated figure that helps users to find appropriate help texts. When a user needs help, the user can query the animated figure and the animated figure can retrieve help text that should answer the user's query. Examples of this type of systems are Microsoft Word's animated figures "Bob" and "the Pin." One major drawback of this type of systems is that services provided are static and do not improve over time. Furthermore, services provided by this type of systems are generic and not personalized to each user.

Another existing system provides so-called "screenmates" that appear on a user's computer screen and perform a limited number of multimedia tasks (e.g., animation or singing a song). Example of this type of systems can be found at: www.messagemates.com and www.oska.com. At messagemate.com, animated figures can be downloaded to perform animations and deliver preprogrammed messages from a server. The animated figures do not stay on the user's computer screen but disappear after they completed the animations and delivered the messages. At oska.com, only one animated figure is downloadable. The oska animated figure does stay on the user's computer screen and randomly performs preprogrammed animations. The screenmates at these web sites have limited capabilities and are not

linked to a server to continuously improve their services to users. Furthermore, these screenmates usually have very limited or no customizable features.

Additional prior art which provides relevant background information in the context of this invention can be found, for example, in *Agent Sourcebook: A Complete Guide to Desktop, Internet, and Intranet Agents*, Colin Harrison and Alper Caglayan (1997); and U.S. Pat. Nos. 5,740,549; 5,890,152; 5,886,698; 5,864,343; 5,774,128; 5,724,567; and 5,263,167, the content of which is incorporated herein by reference.

Thus, it is desirable to provide methods, systems, and software products that leverage the resources of the network (including resources on server computers and client computers) and emphasize intuitive interface and humanized interactions with end users to continuously improve and adapt to end user's preferences, behavior and habits.

Summary of the Invention

This invention comprises methods, systems and software products for providing personalized services to users in a computer network environment. Although the word "personalized" is used herein, this invention is not limited to personalizing services to individuals; this invention can also provide personalized services to groups, families, corporations, organizations, etc. Furthermore, "personalize", "customize", and other similar words are used interchangeably herein.

An exemplary method for providing personalized services to users in a computer network environment comprises the steps of collecting user information through at least one user interface, analyzing the collected user information, generating at least one rule based on the analysis, and providing at least one personalized service to the user based on the generated at least one rule. In one embodiment, the exemplary method also comprises the steps of sending the collected user information to a server for analysis, receiving a response from the server based on the server analysis, and processing the received response. In an exemplary embodiment, the processing step comprises saving the response in a local cache and implementing the response. In another embodiment, the processing step comprises ignoring at least a portion of the response. In an exemplary embodiment, the analyzing step comprises analyzing the user's pattern of behavior. In a preferred embodiment, the at least one user interface is controlled by a software product executing on the user's computer.

In an exemplary embodiment, providing at least one personalized service to users comprises enhancing the user's connection at a web site. The enhancing step comprises creating a separate window when the user visits the web site, the separate window enabling the user to communicate with other users visiting the web site. In one embodiment, the user can communicate with other users in real time.

In another exemplary embodiment, providing at least one personalized service to users comprises creating a connection between a first user and a second user. In one embodiment, the creating step comprises the steps of (i) using at least one icon to carry a message from the first user to the second user; (ii) removing the message carrying icon from the first user's computer screen; (iii) generating a copy of the message carrying icon on the second user's computer screen; (iv) delivering the message to the second user; and (v) establishing a connection between the first user and the second user. In one embodiment, the generating step in (iii) occurs substantially concurrently with the establishing step in (v). In an exemplary embodiment, the establishing step comprises connecting the first user and the second user through a server on the network. In another exemplary embodiment, the establishing step comprises making a direct connection between the first user and the second user.

Another exemplary method for providing personalized services to users in a computer network environment comprises the steps of receiving user information collected by a software product at a user computer, analyzing the collected user information, generating a response based on the analysis and enabling the software product to provide at least one personalized service to the user by sending the generated response to the software product. In one embodiment, the response comprises logic rules, instructions, and/or executable programs.

In an exemplary embodiment, user information is distributed among different computers for analysis. In one embodiment, both a software product on the user's computer and a server communicating with the software product perform the analysis. Analysis performed includes analyzing the user's pattern of behavior. In one embodiment, within the software product, analysis of user information and generation of rules based on the analysis are performed in a multi-layered manner. In another embodiment, the software product executes a high-level command received from the server by adjusting services provided to the user based on realtime interaction with the user.

Yet another exemplary method for providing personalized services to users in a computer network environment comprises the steps of collecting user information through at least one user interface, sending the collected user information to a server for analysis, receiving a response based on the server analysis, and processing the received response to provide at least one personalized service to the user. In one embodiment, the processing step comprises saving the response in a local cache and implementing the response.

Another exemplary method for providing personalized services to users in a computer network environment comprises the steps of communicating with a user through at least one user interface and manipulating the at least one user interface. The manipulating step comprises: (i) automatically initiating the manipulation, (ii) receiving, during the communication, user commands to manipulate the at least one user interface, and (iii) receiving directives from a server to manipulate the at least one user interface. In one embodiment, the manipulating step comprises enabling the at least one user interface to perform animation, enabling the at least one user interface to vary in size and/or enabling the at least one user interface to move from a first location to a second location on a display.

An exemplary computer readable medium for providing personalized services to users in a computer network environment comprises logic code for collecting user information through at least one user interface, logic code for analyzing the collected user information, logic code for generating at least one rule based on the analysis, and logic code for providing at least one personalized service to the user based on the generated at least one rule. In one exemplary embodiment, the computer readable medium also comprises logic code for sending the collected user information to at least one server for analysis, logic code for receiving a response from the at least one server based on the server analysis, and logic code for processing the received response. In an exemplary embodiment, logic code for providing at least one personalized service comprises logic code for providing entertainment to the user and prompting the user to input information. In one embodiment, user interfaces comprise visual characters.

In an exemplary embodiment, the logic code for providing at least one personalized service to users comprises logic code for enhancing the user's connection at a web site. In one embodiment, the logic code for enhancing comprises creating a separate window when the user visits the a web site, the separate window enabling the

user to communicate with other users visiting the web site. In an exemplary embodiment, the user communicates with other users in real time.

In another exemplary embodiment, the logic code for providing at least one personalized service to users comprises logic code for creating a connection between a first user and a second user. In one embodiment, the logic code for creating comprises (i) logic code for using at least one icon to carry a message from the first user to the second user; (ii) logic code for removing the message carrying icon from the first user's computer screen; (iii) logic code for generating a copy of the message carrying icon on the second user's computer screen; (iv) logic code for delivering the message to the second user; and (v) logic code for establishing a connection between the first user and the second user. In an exemplary embodiment, the logic code for generating in (iii) executes substantially concurrently with the logic code for establishing in (v). In one embodiment, the logic code for establishing comprises logic code for connecting the first user and the second user through a server on the network. In another embodiment, the logic code for establishing comprises logic code for making a direct connection between the first user and the second user.

Another exemplary computer readable medium for providing personalized services to users in a computer network environment comprises logic code for receiving user information collected by a software product at a user computer, logic code for analyzing the collected user information, logic code for generating a response based on the analysis, and logic code for enabling the software product to provide at least one personalized service to the user by sending the generated response to the software product. In one embodiment, the response comprises logic rules, instructions, and/or executable programs.

Another exemplary computer readable medium for providing personalized services to users in a computer network environment comprises logic code for logic code for collecting user information through at least one user interface, logic code for sending the collected user information to a server for analysis, logic code for receiving a response based on the server analysis, and logic code for processing the received response to provide at least one personalized service to the user.

Yet another exemplary computer readable medium for providing personalized services to users in a computer network environment comprises at least one user interface and a software product including logic code for controlling the at least one user interface. The logic code for controlling comprises: (a) logic code for

automatically manipulating the at least one user interface, (b) logic code for enabling a user to manipulate the at least one user interface, and (c) logic code for receiving directives from a server for manipulating the at least one user interface. The logic code for controlling comprises logic code for enabling the at least one user interface to perform animation, logic code for enabling the at least one user interface to vary in size, and/or logic code for enabling the at least one user interface to move from a first location to a second location on a display.

An exemplary system for providing personalized services to users in a computer network environment comprises means for collecting user information through at least one user interface, means for analyzing the collected user information, means for generating at least one rule based on the analysis, and means for providing personalized services to the user based on the generated at least one rule. In another exemplary embodiment, the system also includes means for sending the collected user information to at least one server for analysis, means for receiving a response from the at least one server based on the server analysis, and means for processing the received response.

Brief Description of the Drawings

FIGURE 1 schematically illustrates an exemplary system in accordance with an embodiment of the present invention.

FIGURE 2 schematically illustrates an exemplary server used in a system in accordance with an embodiment of the present invention.

FIGURE 3A schematically illustrates an exemplary client in accordance with an embodiment of the present invention.

FIGURE 3B schematically illustrates a portion of an agent program in accordance with an exemplary embodiment of the present invention.

FIGURE 4 is a process flow chart in accordance with an exemplary embodiment of the present invention.

FIGURE 5 is a process flow chart in accordance with another exemplary embodiment of present invention.

FIGURE 6 is a process flow chart in accordance with another exemplary embodiment of present invention.

FIGURE 7 is a process flow chart in accordance with another exemplary embodiment of present invention.

Detailed Description of Exemplary Embodiments

Figure 1 schematically illustrates a distributed system 100 in accordance with an exemplary embodiment of the present invention. The system 100 includes a communications network 102, a server 104, a client 106, other servers 108, other clients 110 and other networks 112. The client 106 includes an agent program 114 which is capable of communicating with the server 104 directly or through the network 102. In a typical network environment, more than one client, server, and network are connected by a common network, such as the communications network 102. In an exemplary embodiment, the communications network 102 is the Internet. Users at client computers (i.e., client 106) can access servers (i.e., server 104) on the network 102.

A person of skill in the art would appreciate that the methods, systems and software products of this invention can be implemented in any network environment and are not limited to a particular physical network architecture. For example, a client computer may be connected to a server computer through an analog modem over a telephone line, a digital subscription line, a cable modem, a dedicated data copper wire connection, an optic fibre connection, a wireless connection, a satellite connection, etc. In one embodiment, a packet switching network implementing TCP/IP protocols is preferred. In an exemplary embodiment, the methods, systems and software products of this invention can be implemented in a wireless network using wireless connection and wireless protocols.

Figure 2 schematically illustrates an exemplary embodiment of the server 104. The server 104 includes CPUs 202, a communication interface 204 for connecting to the network 102 (Fig. 1), and memory 206. In an exemplary embodiment, the memory 206 includes an operating system 208, server applications 210, communications applications 212, an objects database 214, an agent database 216, a user database 218, a user information analyzer 220, a statistics analyzer 222, a collaborative filter 224, a response generator 226, a logic rules database 228 an instructions database 230, a programs database 232, an advertisement database 234, and a recommendations database 236. The server 104 performs server operations by using the server applications 210 and communicates with other systems on the network 102 by using the communication applications 212 through the communication interface 204.

In the objects database 214, the server stores objects downloadable by clients through the network 102. These objects typically have at least one user interface with

the users. In the agent database 216, the server stores downloadable agent programs that maintain any downloaded objects and provide communication with the server 104. In an exemplary embodiment, when a user downloads an object through the client 106, a corresponding agent program is also downloaded simultaneously. Each object is maintained by an associated agent program. However, each agent program is, preferably, capable of maintaining multiple objects. Thus, if an agent program is able to maintain all of the downloaded objects, only one copy of the agent program should be downloaded. In an exemplary embodiment, a downloaded agent program collects user information through at least one object, analyzes the collected user information, and implements any rules generated as a result of the analysis to provide better, more personalized, services to the user. In another exemplary embodiment, the agent program transfers at least a portion of the collected user information to the server 104 to be analyzed, receives any responses from the server 104 based on the server's analysis, and processes the received responses to provide more personalized services to the user.

Referring back to the server 104 in Figure 2, when the server 104 receives user information from an agent program, the server 104 stores the user information in the user database 218. In addition, the server 104 analyzes the received user information using various tools including the user information analyzer 220, the statistics analyzer 222 and the collaborative filter 224. In an exemplary embodiment, the statistics analyzer 222 compares the user information to statistics gathered from other users to deduce personalized services which might interest the user. The collaborative filter 224 can be used to synthesize common patterns and behaviors. In an exemplary embodiment, the collaborative filter may be a licensed software incorporated into the server 104. Based on the analysis, the server 104 may generate a response using the response generator 226. The response generator 226 may generate the response by accessing the logic rules database 228, the instructions database 230, the programs database 232, the advertisement database 234 and/or the recommendations database. In an exemplary embodiment, the server 104 may send a response, which includes one or a combination of logic rules, instructions, executable programs, advertisements, and recommendations. For example, a list of web sites that might interest the user 106 can be retrieved from the recommendations database 236 and sent to the agent 316 to be presented to the user at client 106. A person of skill in the art would recognize that the

tasks described above can be performed by one server computer or by multiple server computers.

Figure 3A schematically illustrates an embodiment of the client 106 (as shown in Fig. 1). Client 106 includes a CPU 302, a display device 304, an input device 306, a communication interface 308 and memory 310. The memory 310 includes an operating system 312, a browser program 314 and an agent program ("agent") 316. The client 106 communicates with other systems on the network 102 (see Fig. 1) through the communication interface 308. In an exemplary embodiment, the client can use the browser program 314 to more efficiently navigate across the network 102.

In various embodiments of this invention, a client may be a personal computer, a workstation, a multiprocessor server, an intelligent appliance, a webTV, a handheld computer device, a cellular phone, or others. In an exemplary embodiment, the client is a cellular phone with an embedded system. The cellular phone is capable of establishing a wireless Internet connection with a server.

In an exemplary embodiment, the agent 316 is a software program downloaded from the server 104 through the network 102 (see Fig. 1). However, the agent 316 may also be pre-installed on a user computer. The agent 316 includes an objects database 318, communication applications 324, a personality database 326, a logic rules database 328, a multimedia database 330, an instructions database 332, a user configuration preference file 334, a user profile database 336, a data analyzer 338, a brain object 340, an object controller 342, and a web browser connector (WBC) object 344. In an exemplary embodiment, the objects database 318 includes user interfaces 322. In one embodiment, the user interfaces include at least one visual interface 322A, at least one audio interface 322B, and at least one multimedia interface 322C. The agent 316 maintains and controls any object in the objects database 318 by supplying each object with personality from the personality database 326, logic rules from the logic rules database 328, multimedia from the multimedia database 330, and instructions from the instructions database 332. Any user configuration preference is saved in the user configuration preference file 334. User information collected by the agent 316 is saved in a user profile database 336. In an exemplary embodiment, the agent 316 uses the data analyzer 338 to analyze any collected user information.

In an exemplary embodiment, the user interfaces 322 as appeared on the user's display device 304 can be manipulated. Examples of manipulation include performing animation, reducing in size, increasing in size, or moving from one location to another

location on the user's display device 304. In an exemplary embodiment, the user interfaces 322 can be manipulated by one or more of the following: (1) the agent 316 may include internal logic code to automatically manipulate the user interfaces 322; (2) the agent 316 may implement user commands to manipulate the user interfaces 322; and (3) the agent 316 may receive directives (e.g., instructions, logic rules, executable programs, etc.) from the server 104 to manipulate the user interfaces 322.

When the client 106 maintains connection to the network 102 using the browser program 314, the WBC object 344 provides the communication channel between the agent 316 and the browser program 314 on the client computer. In an exemplary embodiment, the WBC object 344 is created using Microsoft's Dynamic Data Exchange (DDE) standard. In another embodiment, the WBC object 344 is created using Microsoft's Component Object Model (COM) standard. A person of skill in the art would appreciate that other suitable standards may also be used to create the WBC object 344. When the browser program 314 is used, the agent 316 can communicate to the server 104 through the browser 314 using the WBC object 344 to coordinate the communication with the browser 314. In addition, the agent 316 communicates with the server 104 through the network 102 using the communication applications 322. In an exemplary embodiment, the agent 316 can send any collected user information to the server 104 for analysis. If any user information is sent to the server 104, typically, the server 104 would send back a response which can include logic rules, instructions and/or programs, depending on the server's analysis of the user information. When the agent 316 receives the server response, the agent 316 processes the response by using the brain object 340. In an exemplary embodiment, the object controller 342 implements any brain object 340 process results. In an exemplary embodiment, the brain object 340 and object controller 342 are created using C++ language. These objects can also be created using other suitable computer languages.

Figure 3B schematically illustrates an exemplary embodiment of the relationship between a portion of the agent 316 and objects 350-354 retrieved from the objects database 318 and presented to the users through the user interfaces 322. The agent 316 controls the objects 350, 352 and 354. The agent 316 includes communication applications 324, brain object 340 and object controller 342. In an exemplary embodiment, user interfaces 322 include visual, audio and multimedia interfaces. The agent 316 collects user information through the objects 350-354. In

one embodiment, at least a portion of the collected user information is analyzed by the brain object 340. In another embodiment, a portion or all of the user information is sent to the server 104 to be analyzed. If the brain object 340 analyzes at least a portion of the collected user information, the brain object 340 may generate or retrieve from databases a set of rules or instructions based on its analysis. In an exemplary embodiment, the brain object 340 implements the generated or retrieved set of rules and/or instructions on the objects 350-354 through the object controller 342.

In another exemplary embodiment, the brain object 340 may instruct the communication applications 324 to send the collected user information to the server 104 through the network 102 (Fig. 1). In this embodiment, the communications application 324 may receive responses from the server 104, which could contain instructions, logic rules and/or executable programs. The server response is sent to the brain object 340 through the object controller 342 for processing. In an exemplary embodiment, the brain object 340 processes the server response to reduce it to executable instructions and rules. The brain object 340 sends the executable instructions and rules back to the object controller 342, which administers the implementation of the instructions and rules on the objects 350-354. In an exemplary embodiment, the brain object 340 stores any generated or received instructions and rules in appropriate databases (see Fig. 3A). An example of the source code for the communication applications 324, the brain object 340, and the object controller 342 in C++ language is included in Appendix A.

Figure 4 is a process flow chart illustrating an exemplary process of the present invention. In this embodiment, a user downloads at least one object and an agent program through a client computer from a server. The server can be the exemplary server 104 or any other servers on the network 102. Likewise, the agent program can be the exemplary agent 316 or other downloadable agent programs. The object(s) downloaded is controlled by the agent program. The agent program ("agent") communicates with the user through at least one user interface of the object(s) (block 402). During communication, the agent collects user information (block 404). For example, the user may be prompted by the object's visual interface on the computer screen to answer a specific question. In a preferred embodiment, any input from the user is collected by the agent and saved in a user profile database (block 404A). In an exemplary embodiment, the agent analyzes the collected user information (block 406). Based on the analysis, the agent generates rules and/or instructions (block 408). The

agent implements the generated rules and/or instructions on the object(s) to provide the user with improved personalized services (block 410). For example, if during the analysis, the agent learns that the user is interested in going to a specific amusement park, the agent may instruct the object(s) (through its visual, audio or multimedia interfaces) to provide vacation packages or other information regarding that amusement park.

Figure 5 is a process flow chart illustrating an exemplary process of the present invention. A user downloads at least one object and an agent through a client computer from a server. The downloaded object(s) is controlled by the downloaded agent. The agent can be the exemplary agent 316 or any other downloadable agents. The agent communicates with the user through at least one user interface (visual, audio or multimedia interfaces) of the object(s) (block 502). During communication, the agent collects user information (block 504). In an exemplary embodiment, any input from the user is collected by the agent and stored in a user profile database (block 504A). The agent then substantially concurrently analyzes the collected user information (block 506) and sends the collected user information to a server, such as the exemplary server 104, for analysis (block 510). The agent generates a set of rules and instructions based on its own analysis (block 508) and receives a response from the server based on the server's analysis (block 512). In one embodiment, the agent first implements the generated rules and/or instructions and then the server response to personalize services to the user. In another embodiment, the agent first implements the server response then the rules and instructions generated based on its own analysis. The implementation sequences can be programmed by the server or service provider.

Figure 6 is a process flow chart illustrating an exemplary process of the present invention. A user downloads at least one object and an agent through a client computer from a server. The server can be the exemplary server 104 or any other servers on the network 102. Likewise, the agent program can be the exemplary agent 316 or other downloadable agent programs. The downloaded object(s) is controlled by the downloaded agent. The agent communicates with the user through at least one user interface (visual, audio or multimedia interfaces) of the object(s) (block 602). During communication, the agent collects user information (block 604). Input from the user is collected by the agent and stored in a user profile database (block 604A). The agent sends the collected user information to a server for analysis (block 606). The agent receives a response from the server based on the server's analysis of the user

information (block 608). The agent processes and implements the response to personalize services provided to the user (block 610).

Figure 7 is a process flow chart illustrating an exemplary process in accordance with another embodiment of the present invention. A server receives user information from an agent, which was downloaded to the user's computer (block 702). The server analyzes the received user information (block 704). In an exemplary embodiment, the server saves the received user information in a user database (block 704A). Based on the analysis, the server generates a response which may include instructions, logic rules and/or executable programs (block 706). The server sends the generated response to the agent 316 to be processed and implemented by the agent to personalize services provided to the user (block 708).

General Operation

In an exemplary embodiment, an object's user interface may appear to a user as an animated figure having multimedia capabilities. From the user's point of view, the user is getting a desktop companion, who is capable of realtime interaction and becomes more "intelligent" as it learns more about the user. In an exemplary embodiment, the animated figure may automatically appear on the user's computer screen each time the user starts the computer. While the user's computer is running, the animated figure remains on the desktop and can be commanded to sit at a corner, to roam around the screen, or to disappear. The animated figure is maintained and controlled by an agent program. In one embodiment, as the animated figure collects user information (i.e., by prompting the user to answer questions), the collected user information is processed by the agent program at the user computer. In another embodiment, a portion or all of the collected user information can be sent to a server to be analyzed. The information is used to improve personalized services to the user so that, in some embodiments, the animated figure appears to have gained "intelligence" over time. For example, from time to time, the agent may enable the animated figure to prompt the user with personal questions to find out more about the user's interests. For instance, the animated figure may say to the user, "I like baseball, do you?" If the user answers "yes", the agent will provide more information about baseball. Likewise, if the user answers "no", the agent will not bring up baseball again. Thus, the animated figure appears to the user as having the capability of gradually improving its "intelligence" because of the continuous analysis of collected user information, either

by the agent program 316 or by the server 104. In an exemplary embodiment, each animated figure has its own pre-assigned personalities and an initial repertoire of multimedia capabilities (e.g., sing, dance, speak, etc.). As the animated figure learns more about a user, the behavior of the animated figure at that user's computer can be personalized to that user's taste. Examples of presentations by animated figures to users are included in Appendix B.

In an exemplary embodiment, the animated figure may communicate with the user through audio, video, animation, or other multimedia interfaces. The user can control the animated figure by using voice commands or from a menu selection.

In an exemplary embodiment, when a user selects an object (e.g., an animated figure) to download, an agent program associated with the object is downloaded to the user computer. If the user already has an agent program that is capable of controlling the selected object, then the agent program associated with the selected object is not downloaded. However, a user's computer may have multiple agent programs. For example, if the selected object requires a different agent program to maintain it, that agent program should be downloaded with the selected object even if the user already has another agent program on his computer.

A person skilled in the art would recognize that the visual interfaces of the objects are not limited to animated figures. Visual interfaces may come in a variety of images, such as animals, fantasy cartoon figures, or other suitable images. The images can also be custom created by each user. In an exemplary embodiment, depending on the image, a suitable pre-determined personality should be assigned.

Embodiments of this invention provide a wide range of personalized services. Examples of the types of personalized services include: entertainment related, communication related, electronic commerce related, education related services, and others. One example of entertainment related services include periodic delivery of games to a user who enjoys trying new games. In another example, the objects may execute simple programs, such as singing a song, dancing a routine, telling a riddle or a story to the user.

An example of communication related services is carrying a user's bookmark from one computer on the network to another computer on the network. Another example of communication related services is creating a separate window at a third party web site, which enables several users to communicate with each other real time. In addition, a separate window created at the third party web site may enable each user

visiting the web site to post comments about the web site. Thus, each user having an agent program will be able to see the separate window when the user visits the third party web site. Other communication related services include enhancing the connection between users on the network, such as improving speed and reliability. In addition, the connection between users on the network may allow realtime exchange of multimedia contents.

Another example of communication related services is establishing a connection between two users. In this example, a first user sends a message to a second user on the network. The agent receives the message from the first user through the object(s). The agent removes a message carrying icon from the first user's computer screen and generates a copy of the removed icon on the second user's computer screen. As the icon is generated at the second user's computer screen, the agent delivers the message to the second user and establishes a connection between the first user and the second user. Thus, it may appear to the users that the connection between them was achieved by the icon when in reality, the connection may be performed by the server. The message transferred between users can include textual, audio, video or other multimedia contents.

In another communication related service, the agent can use the collected user information to automatically conduct an efficient search on the Internet using existing search engines. Because the agent has collected user information, it is able to conduct an efficient search for the user based on those collected information. In yet another communication related service, the agent can create a separate window which the user can use as a note pad. For example, if a user is researching a certain subject matter on the Internet, he is likely to move from web site to web site. At each web site, the user may wish to take notes in the created separate window. The notes written by the user can be collected by the agent and analyzed (by the agent and/or a server) to further improve services provided to the user.

An example of electronic commerce related services is providing a visual interface, which introduces the latest fashion to a user. Businesses in the fashion industry may be interested in providing a desktop model, which maintains communication with the business's server and continually changes clothing style to promote the business's products. In this example, the agent can continuously learn about the user's taste in fashion and personalize the presentation of the business's products. This embodiment is, of course, not limited to only the fashion industry.

Other businesses can also implement this embodiment to promote and sell their products. In another embodiment, the agent, through a user interface, can present product advertisement that might be of interest to a user and can direct the user to a commercial web site.

An example of education related services is using the agent as a tutor for a specific subject matter. In this embodiment, the agent provides a user interface to assist a user's educational needs. The agent communicates with a server that maintains and updates educational subject matters (e.g., math problems and solutions). As the agent continues to learn more about the user and the subject matter in which the user needs the most assistance, the agent can provide more services in that area. In another example, if the agent is part of a proprietary software program, the agent can provide personalized interactive technical support to a user, retrieve specific information for the user from a server, and automatically upgrades the program for the user.

In an exemplary embodiment, the methods, systems and software products of this invention can be created in a C++ programming environment and implemented on a Windows 95/98/NT platform. Such programs can be automatically upgraded from time to time. The methods, systems and software products of this invention can also be created using other programming environments, including: C, Java, Perl, Basic Fortran, COBOL, or other suitable computer languages. In a preferred embodiment, object-oriented programming languages are used. In addition, software programs in accordance with embodiments of this invention are stored in a computer readable medium, such as a computer hard-drive, a CD-ROM, a magnetic tape, a flash memory, or other computer readable media.

Preferably, the agent program should operate without consuming too much memory and CPU time on the user's computer. Coordinating with and transferring some tasks to the server prevent the agent program from overwhelming the user's resources. In an exemplary embodiment, the server is not limited to one computer, it may include as many cooperating servers as needed. The servers can also be custom-designed and controlled by proprietary service providers. Task division among many servers provides more flexible server extensibility and scalability, among other benefits. In an exemplary embodiment, the server can be implemented on a Unix platform and may use a variety of tools, such as C++ and Perl languages, open-source Apache webserver, and relational database (e.g., free MSQL and Oracle). In addition,

the methods, systems and software products in accordance with the present invention should preferably be implemented using multi-threaded technology. Multi-threaded technology allows different programs to be executed independently of each other during a given time period.

The methods, systems and software products of this invention can be implemented in any network environment, and are preferably implemented in a distributed network environment (e.g., the Internet). In an exemplary embodiment, the methods, systems and software products of this invention can be implemented in a wireless network environment (e.g., cellular technologies, handheld device technologies, etc.). Wireless network environment may require computer programs in accordance with embodiments of this invention to be written in programming languages suitable for wireless platforms.

The methods, systems and software products in accordance with various embodiments of this invention provide many advantages for both users and service providers (servers). Users can obtain personalized services in a user friendly and intuitive environment. Service providers can efficiently provide personalized services without spending on excessive advertisement. For example, businesses may adopt the methods, systems and software products to more effectively sell their products. In addition, service-based businesses may profit from the various embodiments of this invention. For example, customized animated figures and their associated agent programs can be created according to embodiments of this invention and sold to target organizations or web sites. Service-based businesses then can profit by maintaining servers which provide continuous improvements to the agent programs and animated figures.

The methods, systems and software products in accordance with various embodiments of this invention overcome many disadvantages in existing systems. Various embodiments of this invention dynamically improves personalized services provided to users. Unlike most existing systems, the present methods, systems and software products, with the user's permission, collect user information in a user friendly, intuitive, and continuously improved manner. Furthermore, in exemplary embodiments of this invention, users always have the option to deny information collection.

The foregoing examples illustrate certain exemplary embodiments of the invention from which other embodiments, variations, and modifications will be

apparent to those skilled in the art. The invention should therefore not be limited to the particular embodiments discussed above, but rather is defined by the following claims.

WHAT IS CLAIMED IS:

1. A method for providing personalized services to users in a computer network environment, comprising the steps of:

(a) collecting user information through at least one user interface;

(b) analyzing said collected user information;

(c) generating at least one rule based on said analysis; and

(d) providing at least one personalized service to said user based one said generated at least one rule.

2. The method of claim 1, further comprising the steps, performed after at least said step (b), of: (e) sending said collected user information to a server for analysis; (f) receiving a response from said server based on said server analysis; and (g) processing said received response.

3. The method of claim 2, wherein said step (g) comprises: (i) saving said response in a local cache; and (ii) implementing said response.

4. The method of claim 2, wherein said step (g) comprises ignoring at least a portion of said response.

5. The method of claim 1, wherein at least one of said step (a) and said step (d) comprises providing entertainment to said user.

6. The method of claim 1, wherein at least one of said step (a) and said step (d) comprises prompting said user to input information.

7. The method of claim 1, wherein said step (c) comprises analyzing said user's pattern of behavior.

8. The method of claim 1, wherein said step (d) is performed substantially concurrently with said collecting in said step (a).

9. The method of claim 1, wherein said step (d) is performed subsequent to said collecting in said step (a).

10. The method of claim 1, wherein said step (d) comprises enhancing said user's connection at a web site.

11. The method of claim 10, wherein said enhancing comprises creating a separate window when said user visits said web site, said separate window enabling said user to communicate with other users visiting said web site.

12. The method of claim 11, wherein said user communicates with other users in real time.

13. The method of claim 1, wherein said step (e) comprises creating a connection between said user and a second user.

14. The method of claim 13, wherein said creating comprises:

(i) using at least one icon to carry a message from said user to said second user;

(ii) removing said message carrying icon from said user's computer screen;

(iii) generating a copy of said message carrying icon on said second user's computer screen;

(iv) delivering said message to said second user; and

(v) establishing a connection between said user and said second user;

wherein said generating in (iii) occurs substantially concurrently with said establishing in (v).

15. The method of claim 14, wherein said establishing in step (v) comprises connecting said user and said second user through a server on said network.

16. The method of claim 14, wherein said establishing in step (v) comprises making a direct connection between said user and said second user.

17. A method for providing personalized services to users in a computer network environment, comprising the steps of:

(a) receiving user information collected by a software product at a user computer;

(b) analyzing said collected user information;

(c) generating a response based on said analysis; and

(d) enabling said software product to provide at least one personalized service to said user by sending said generated response to said software product.

18. The method of claim 17, wherein said response comprises logic rules.

19. The method of claim 17, wherein said response comprise instructions.

20. The method of claim 17, wherein said response comprises executable programs.

21. A method for providing personalized services to users in a computer network environment, comprising the steps of:

(a) collecting user information through at least one user interface;

(b) sending said collected user information to a server for analysis;

(c) receiving a response based on said server analysis; and

(d) processing said received response to provide at least one personalized service to said user.

22. The method of claim 21, wherein said step (d) comprises: (i) saving said response in a local cache; and (ii) implementing said response.

23. A computer readable medium for providing personalized services to users in a network environment, comprising:

(a) logic code for collecting user information through at least one user interface;

(b) logic code for analyzing said collected user information;

(c) logic code for generating at least one rule based on said analysis; and

(d) logic code for providing at least one personalized service to said user based on generated at least one rule.

24. The computer readable medium of claim 23, further comprising: (e) logic code for sending said collected user information to at least one server for analysis; (f) logic

code for receiving a response from said at least one server based on said server analysis; and (g) logic code for processing said received response.

25. The computer readable medium of claim 24, wherein said (g) comprises: (i) logic code for saving said response in a local cache; and (ii) logic code for implementing said response.

26. The computer readable medium of claim 24, wherein said (g) comprises logic code for ignoring at least a portion of said response.

27. The computer readable medium of claim 23, wherein at least one of said (a) and said (d) comprises logic code for providing entertainment to said user.

28. The computer readable medium of claim 23, wherein at least one of said (a) and said (d) comprises logic code for prompting said user to input information.

29. The computer readable medium of claim 23, wherein said (b) comprises analyzing said user's pattern of behavior.

30. The computer readable medium of claim 23, wherein said at least one user interface comprises visual characters.

31. The computer readable medium of claim 23, wherein said (d) comprises logic code for enhancing said user's connection at a web site.

32. The computer readable medium of claim 31, wherein said logic code for enhancing comprises logic code for creating a separate window when said user visits said web site, said separate window enabling said user to communicate with other users visiting said web site.

33. The computer readable medium of claim 32, wherein said user communicates with other users in real time.

34. The computer readable medium of claim 23, wherein said (e) comprises logic code for creating a connection between said user and a second user.

35. The computer readable medium of claim 13, wherein said logic code for creating comprises:

(i) logic code for using at least one icon to carry a message from said user to said second user;

(ii) logic code for removing said message carrying icon from said user's computer screen;

(iii) logic code for generating a copy of said message carrying icon on said second user's computer screen;

(iv) logic code for delivering said message to said second user; and

(v) logic code for establishing a connection between said user and said second user;

wherein said logic code for generating in (iii) executes substantially concurrently with said logic code for establishing in (v).

36. The computer readable medium of claim 35, wherein said logic code for establishing in (v) comprises logic code for connecting said user and said second user through a server on said network.

37. The computer readable medium of claim 35, wherein said logic code for establishing in (v) comprises logic code for making a direct connection between said user and said second user.

38. A computer readable medium for providing personalized services to users in a computer network environment, comprising:

(a) logic code for receiving user information collected by a software product at a user computer;

(b) logic code for analyzing said collected user information;

(c) logic code for generating a response based on said analysis; and

(d) logic code for enabling said software product to provide at least one personalized service to said user by sending said generated response to said software product.

39. The computer readable medium of claim 38, wherein said response comprises logic rules.

40. The computer readable medium of claim 38, wherein said response comprises instructions.

41. The computer readable medium of claim 38, wherein said response comprises executable programs.

42. A computer readable medium for providing personalized services to users in a computer network environment, comprising:

(a) logic code for collecting user information through at least one user interface;

(b) logic code for sending said collected user information to a server for analysis ;

(c) logic code for receiving a response based on said server analysis; and

(d) logic code for processing said received response to provide at least one personalized service to said user.

43. The computer readable medium of claim 42, wherein said (e) comprises: (i) logic code for saving said response in a local cache; and (ii) logic code for implementing said response.

44. A system for providing personalized services to users in a computer network environment, comprising:

(a) means for collecting user information through at least one user interface;

(b) means for analyzing said collected user information;

(c) means for generating at least one rule based on said analysis; and

(d) means for providing at least one personalized service to said user based on said generated at least one rule.

45. The system of claim 44, further comprising: (e) means for sending said collected user information to at least one server for analysis; (f) means for receiving a response from said at least one server based on said server analysis; and (g) means for processing said received response.

46. A computer readable medium for providing personalized services to users in a computer network environment, comprising:

at least one user interface; and

a software product including logic code for controlling said at least one user interface;

said logic code for controlling comprises:

(a) logic code for automatically manipulating said at least one user interface;

(b) logic code for enabling a user to manipulate said at least one user interface; and

(c) logic code for receiving directives from a server for manipulating said at least one user interface.

47. The computer readable medium of claim 46, wherein said logic code for controlling comprises logic code for enabling said at least one user interface to perform animation.

48. The computer readable medium of claim 46, wherein said logic code for controlling comprises logic code for enabling said at least one user interface to vary in size.

49. The computer readable medium of claim 46, wherein said logic code for controlling comprises logic code for enabling said at least one user interface to move from a first location to a second location on a display.

50. A method for providing personalized services to users in a computer network environment, comprising the steps of:

(a) communicating with a user through at least one user interface; and

(b) manipulating said at least one user interface;

wherein said manipulating comprises: (i) automatically initiating said manipulation; (ii) receiving, during said communication, user commands to manipulate said at least one user interface; and (iii) receiving directives from a server to manipulate said at least one user interface.

51. The method of claim 50, wherein said manipulating in said (b) comprises enabling said at least one user interface to perform animation.

52. The method of claim 50, wherein said manipulating in said (b) comprises enabling said at least one user interface to vary in size.

53. The method of claim 50, wherein said manipulating in said (b) comprises enabling said at least one user interface to move from a first location to a second location on a display.

APPENDIX A

Partial Source Code for Object Controller 342:

```
////////////////////////////////////////////////////////////////
// Construction/Destruction
////////////////////////////////////////////////////////////////

CQTCharController::CQTCharController(Cstring root_dir)
{
        hQTCommHWND=NULL;

        clientRootDirroot_dir;
        dataSubDir = clientRootDir + "data\\";
        animationSubDir = dataSubDir + "animation\\";

        for(int i=0; i<MAX_CHARACTER_COUNT; i++){
                own_chars[i]=NULL;
                guest_chars[i]=NULL;
                own_char_brains[i]=NULL;
        }

}

CQTCharController::~CQTCharController()
{
        for(int i=0; i<MAX_CHARACTER_COUNT; i++){
                if( own_chars[i] !=NULL) delete own_chars[i];
                if( guest_chars[i] !=NULL) delete guest_chars[i];
                if( own_char_brains[i] != NULL) delete own_char_brains[i];
        }
}

int CQTCharController::CreateMainWindow()
{
  WNDCLASS wc;
  {
    wc.lpszClassName = "QTCharControllerClass";
    wc.lpfnWndProc = QTCharControlWndProc;
    wc.style =  CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
    wc.hInstance = GetModuleHandle(NULL);
    wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
    wc.hCursor = LoadCursor( NULL, IDC_ARROW );
    wc.hbrBackground = (HBRUSH)( COLOR_WINDOWFRAME );
    wc.lpszMenuName = NULL;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
```

```
    RegisterClass( &wc );
  }

  //handle hThreadHWND declared in CQTThread base class.
  hThreadHWND = CreateWindowEx(WS_EX_TOPMOST,
        "QTCharControllerClass", "QT", WS_POPUP, 300,300,20,20, NULL,NULL,
        GetModuleHandle(NULL), NULL );
      ShowWindow(hThreadHWND,SW_HIDE);
      ::PostMessage(hThreadHWND,WM_USER + CHAR_CONTROL_INIT,0,0); //tell this thread to
initialize itself.
//    SetTimer(this->hThreadHWND,CQT_COMM_TIMER,this->clock_interval, NULL);
      return 1;
}

//////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////


int CQTCharController::ProcessMessage(MSG *msg)
{
      switch( msg->message){

      case WM_USER + CHAR_CONTROL_INIT:
            this->Initialize();   //we put init here instead of the constructor, because we want the main
thread to finish creating all the thread
                                                                  //ASAP, instead of spending time
initializing one particular thread while blocking
                                                                  //the synchronization of all the threads.

            //if we need user to login, don't do anything here, the Command Center will take user input,
            // configure the logged in userID & charID, and then post a message;
            //
            //otherwise (also for first version), login default userID & charID:
            ::PostMessage(this->hThreadHWND, WM_USER + USER_COMMAND,
USER_COMMAND_LOGIN,0);
      case WM_TIMER:
            if( msg->wParam == WM_USER /*TICK*/){
                  //PBrain->Tick();
            }
            break;


      case WM_USER + USER_COMMAND: //commands from Command Control Center;
            if( msg->wParam == USER_COMMAND_LOGIN){

                  //load char for charID; load brain with charID & userID:
                  //own_chars[0] = charServer->LoadCharacter(this->CurrentCharID);
                  own_chars[0] = charServer->LoadCharacter("jc");
                  own_char_brains[0] = new CQTBrain();
```

```
                        //connect brain & char, now brain will drive char's behavior.
                        own_char_brains[0]->Initialize(this->CurrentUserID, own_chars[0]);
                }
                break;
        case WM_USER + 1001://debug test
                own_chars[0]->Show();
                own_chars[0]->Play("hello");
                break;
        default:
                return 0;
        }
        return 1;
}

////////////////////////////////////////////////////////////////////////////////////////
//// called after the first message this object receives, sent from CreateMainWindow().

int CQTCharController::Initialize()
{
        //create CommandController, ask for login if so configured; try not to block here.


        //create Character Server;
        charServer = new CQTCharServer(this->animationSubDir);

        //create Resource Server object, start its thread;
        return 1;
}
```

Partial Source Code for Communication Applications Object 324:

```
//////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////

CQTComm::CQTComm()
{
 hQTCharControllerHWND=NULL;
 server_response_buf=(char*) malloc(INCOMING_BUF_SIZE+1);
 servers=new MountainServer("everest",80,DEFAULT_SERVER);
 clock_interval=CQT_COMM_TIMER_DEFAULT;
 in_buffer=new CQTCommBuffer();
 out_buffer=new CQTCommBuffer();

 hQTCharControllerHWND = NULL;
}

CQTComm::~CQTComm()
{
        free(this->server_response_buf);
        if( in_buffer) delete in_buffer;
        if( out_buffer) delete out_buffer;
}

int
CQTComm::ProcessMessage(MSG* msg)
{
int IncomingBufferIndex=-1;
int OutgoingBufferIndex=-1;

        switch(msg->message){
        case WM_TIMER: //sent from the clock
                IncomingBufferIndex=ReportToServer(); //this line will fill an incoming buffer;
                HandleServerCommand(IncomingBufferIndex);  //this line will process and have it freed
                return 1;

                        //the Brain may ask comm to send data up:
        case (WM_USER + COMM_SEND_DATA_TO_SERVER):
                OutgoingBufferIndex=LOWORD(msg->wParam);
                IncomingBufferIndex=SendDataToServer(OutgoingBufferIndex);//this line will find a free
incominb buffer, and fill it.
                out_buffer->CleanBuffer(OutgoingBufferIndex); //clean the outgoing buffer once it is sent.
                HandleServerCommand(IncomingBufferIndex); //this line will process and have the incoming
buffer freed
                return 1;
        }
        return 0;
```

```
}

// this method is invoked in base class CQTThread::StartThread, the thread will enter message loop right
// after this function call.
int CQTComm::CreateMainWindow()
{
  WNDCLASS wc;
  {
    wc.lpszClassName = "QTCommClass";
    wc.lpfnWndProc = QTCommWndProc;
    wc.style =  CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
    wc.hInstance = GetModuleHandle(NULL);
    wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
    wc.hCursor = LoadCursor( NULL, IDC_ARROW );
    wc.hbrBackground = (HBRUSH)( COLOR_WINDOWFRAME );
    wc.lpszMenuName = NULL;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;

    RegisterClass( &wc );
  }

  //handle hThreadHWND declared in CQTThread base class.
  hThreadHWND = CreateWindowEx(WS_EX_TOPMOST,
          "QTCommClass", "JC", WS_POPUP, 300,300,20,20, NULL,NULL,
           GetModuleHandle(NULL), NULL );
        ShowWindow(hThreadHWND,SW_HIDE);

        //to_do: add error checking here - if failed to create window, we'll report to server.

              //CQTComm thread needs to have a clock to periodically
              //communicate with JJ Mountain Servers.
        SetTimer(this->hThreadHWND,CQT_COMM_TIMER,this->clock_interval, NULL);
        return 1;
}


LRESULT WINAPI QTCommWndProc( HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam )
{
  switch( msg ) {
    case WM_DESTROY:
     PostQuitMessage( 0 );
      break;
    default:
      return( DefWindowProc( hWnd, msg, wParam, lParam ));
  }
  return 0;
}
```

```
////////////////////////////////////////////////////////////////////////
int CQTComm::ReportToServer()
{
        int buf_index=out_buffer->GetFreeBuffer();

        char buf[124];
        sprintf(buf,"name=%s&friend=%s", "jc","cc");

        out_buffer->AppendToBuffer(buf_index,buf,strlen(buf));
        return this->SendDataToServer(buf_index);
}

////////////////////////////////////////////////////////////////////////
int CQTComm::SendDataToServer(int out_buffer_index, int server_index)
{
        int result=-1;
        //find out which server to send to:
        MountainServer* mserver=this->GetServer(server_index);

        if( !mserver) {
                MessageBox(NULL,"exception COMM01- Pet Can't find server. Please email
bug@jjmountain.com","COMM",MB_OK );
                return 1;
                }

        CString strServerName=mserver->GetName();
        INTERNET_PORT nPort=mserver->GetPort();


        //connect to server:
        CInternetSession session("mysession");
        CHttpConnection* pServer = NULL;
        CHttpFile* pFile = NULL;
        pServer = session.GetHttpConnection(strServerName, nPort);

        //Send to server:
        char buf[124];
        LPCTSTR ss=out_buffer->GetBufferAddress(out_buffer_index);

//      char* s;
        sprintf(buf,"/demo/jc.asp?%s",ss);
        ss=buf;


        //pFile2->AddRequestHeaders(szHeaders);
        try{
                pFile = pServer->OpenRequest(CHttpConnection::HTTP_VERB_GET,
ss,NULL,1,NULL,NULL,INTERNET_FLAG_RELOAD);
                pFile->SendRequest();
```

```
}catch(CInternetException e){
                MessageBox(NULL,"exception","COMM",MB_OK );
                delete pFile;
                delete pServer;
                session.Close();
                out_buffer->CleanBuffer(out_buffer_index);
                return -1;
}
        out_buffer->CleanBuffer(out_buffer_index);

        DWORD dwRet;
        pFile->QueryInfoStatusCode(dwRet);

        int total_read=0;
        memset(server_response_buf,0,2048);
        if (dwRet == HTTP_STATUS_OK)    {
          UINT nRead = pFile->Read(this->server_response_buf, 2048);
                this->server_response_buf[nRead] = '\0';
                total_read=nRead;
        }else{
                delete pFile;
                delete pServer;
                session.Close();
                return -1; //bad connection? disrupted stream? whatever.
        }

        //is there more to read?
        int content_length=128; //to_do: decode server_response_buf to find content-length;


        if(1 || total_read == content_length){ //OK, no more contents to read:
                int inbuffer_index=in_buffer->GetFreeBuffer();
                in_buffer->AppendToBuffer(inbuffer_index, this->server_response_buf, total_read);
                //error checking?
                result=inbuffer_index;
        }else if(total_read < content_length && content_length < 2048){
                //error: failed to read everything -- can this actually happen?

                result = -1;
        }else{//well, seems we needa larger buffer:
                int inbuffer_index=in_buffer->GetFreeBuffer( content_length);
                in_buffer->AppendToBuffer(inbuffer_index, this->server_response_buf, total_read);
                int nRead=pFile->Read(this->server_response_buf,1023); //read file...
                total_read += nRead;
                while (nRead > 0)
                {
                        in_buffer->AppendToBuffer(inbuffer_index, this->server_response_buf,
nRead);
                        nRead=pFile->Read(this->server_response_buf,1023); //read file...
```

```
                    total_read += nRead;
                }

                if( total_read < content_length){ //error: interrupted?
                    in_buffer->CleanBuffer(inbuffer_index);
                    result = -1;
                }else{
                    result =inbuffer_index;
                }
            }

            delete pFile;
            delete pServer;
            session.Close();

            return result;
}


////////////////////////////////////////////////////////////////////////////
int CQTComm::HandleServerCommand(int IncomingBufferIndex)
{

    if(in_buffer->OutOfBound(IncomingBufferIndex)) return 0;
    LPCTSTR buf=in_buffer->GetBufferAddress(IncomingBufferIndex);
    if(!buf) return 0;

    if(1 || !strcmp(buf, "COMMCOMMAND")){
        //process;
        CString msg;
        char string_buf[20];
        sprintf(string_buf, "buffer: %d", IncomingBufferIndex);
        msg=string_buf;
        MessageBox(NULL,"received command." + msg,"COMM",MB_OK );
        in_buffer->CleanBuffer(IncomingBufferIndex);

    }else{
        WPARAM wParam=MAKEWPARAM(0,IncomingBufferIndex);
        if( this->hQTCharControllerHWND != NULL) {
            ::PostMessage(this->hQTCharControllerHWND,
WM_USER+WM_SERVER_COMMAND, wParam,0);
            //the receiving thread will take the responsibility of freeing the income buffer.
        }else{ //sleep for 1 second, try again:
            Sleep(1000);
            if( this->hQTCharControllerHWND)
                ::PostMessage(this->hQTCharControllerHWND,
WM_USER+WM_SERVER_COMMAND, wParam,0);
            else{
                //if that still fails, report to the server, or ignore.
```

```
                    }
                }
            }
            return 1;
        }
```

```
int CQTComm::ThreadCleanUp() //implementation of virtual function defined in base class CQTThread;
{
        if( this->hThreadHWND)
                KillTimer(this->hThreadHWND,CQT_COMM_TIMER);
        return 1;
}
```

Partial Source Code for Brain Object 340:

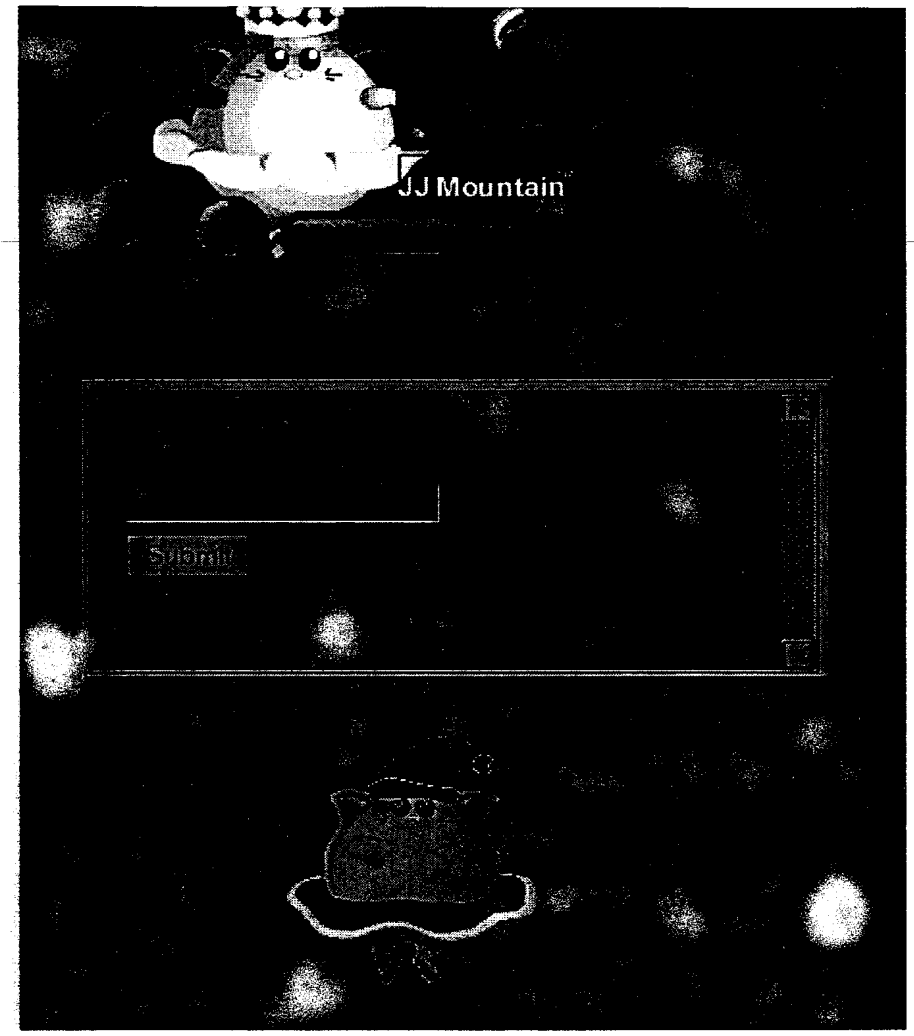
```
// QTBrain.h: interface for the CQTBrain class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_QTBRAIN_H__83213185_941C_11D2_9641_00105A9E5870__INCLUDED_)
#define AFX_QTBRAIN_H__83213185_941C_11D2_9641_00105A9E5870__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "QTHostSystem.h"
#include "QTScriptEngine.h" // Added by ClassView
#include "QTBrainCommon.h"
#include "QTBrainImageMap.h"
#include "QTCommand.h"
#include "QTTicGame.h"

class CQTCharacter;
class CQTCharController;
class CQTDisplay;
class CQTDisplayController;
class CQTCommandMessenger;
class CQTCommandChat;
class CQTCommandEmail;

class CQTKnockoutScoreboard;


class CQTBrain : public CQTThread
{

public:

        int HServerExecute(CString scriptName);

        int HHandleServerCommand(MSG* msg);
        int HUserShowGreeting(CString greeting_name);
        int HUserPlayGame(CString game_name);
        int HUserSendEmail();
        int HUserOpenEmailWindow();
        int CatchMeGame();
        int ActionInit();
        int CheckIdleActivity();
        int UserDragged();
        int UserClickMessage(MSG* msg,int petType);
        int UserTryMultiInstance();
        int AddShortcut(int addOrRemove);
```

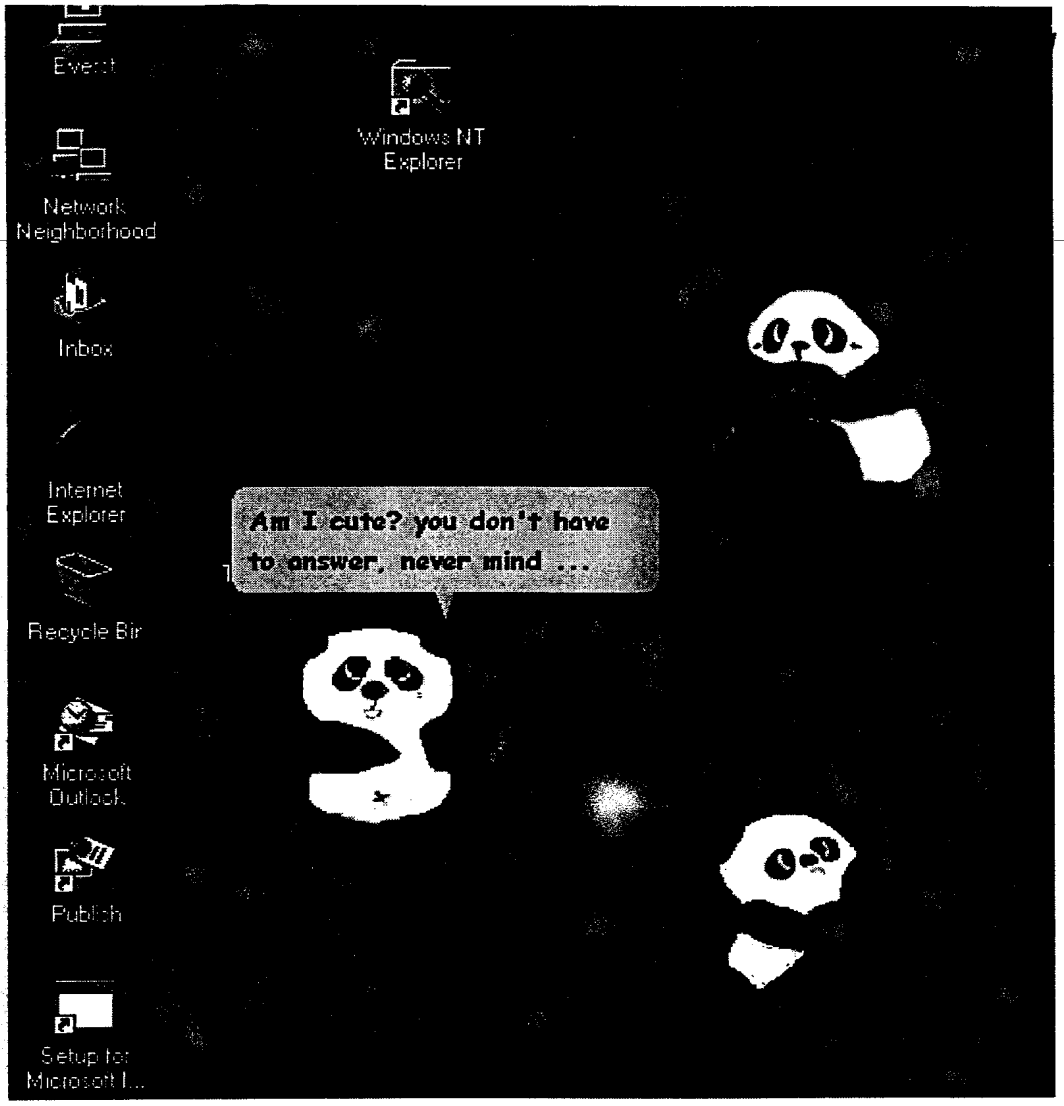
```
CString dataDir;
int SaveResourceToFile(int rcID, LPCTSTR rcType);
int CheckCarriedData();
int PlayMIDI(LPCTSTR fileName);

//////////////////////////////////////
///////USER INPUT API:
int UserRightClicked();
int UserLeftDoubleClicked();
int UserRightButtonMove();


//Internal Behavior Functions:
int Tick(MSG* msg);  //the clock ticks, brain needs to figure out what to do and do it;
                                    //This function blocks -- it doesn't return until it does whatever it is
supposed to do.
                                    // Brain uses a state machine to manage what to do next;

int Pause(); //pause execution of scriptEngine
int Resume();
void AllowInterrupt(bool choice){ if(choice) this->allowInterrupt=1; else this->allowInterrupt=0;}

int StartDefaultBrowserWith(HWND hWnd, LPCTSTR lpURL);


//Internal State Functions:
void   SetState(DWORD state){brainState = brainState | state;}
void  ResetState(DWORD state){brainState = brainState & ~state;}
bool    IsState(DWORD state){return ((brainState & state) == state);}

// INTERNAL FUNCTIONS:
int CreateMainWindow();
int Initialize(CQTCharacter* pChar, CQTCharController* pCon); //executed on calling thread;
                                        //local thread initialization done in ActionInit()
int ThreadCleanUp();
int ProcessMessage(MSG* msg);
int ProcessComic(MSG* msg);
int ProcessChat(MSG* msg);
int ProcessGame(MSG* msg);     //should be challenging, with various levels;
int ProcessMessenger(MSG* msg); //should be able to preview, should have a library of animations.


//CONSTRUCT DESTRUCT:
CQTBrain();
virtual ~CQTBrain();
```

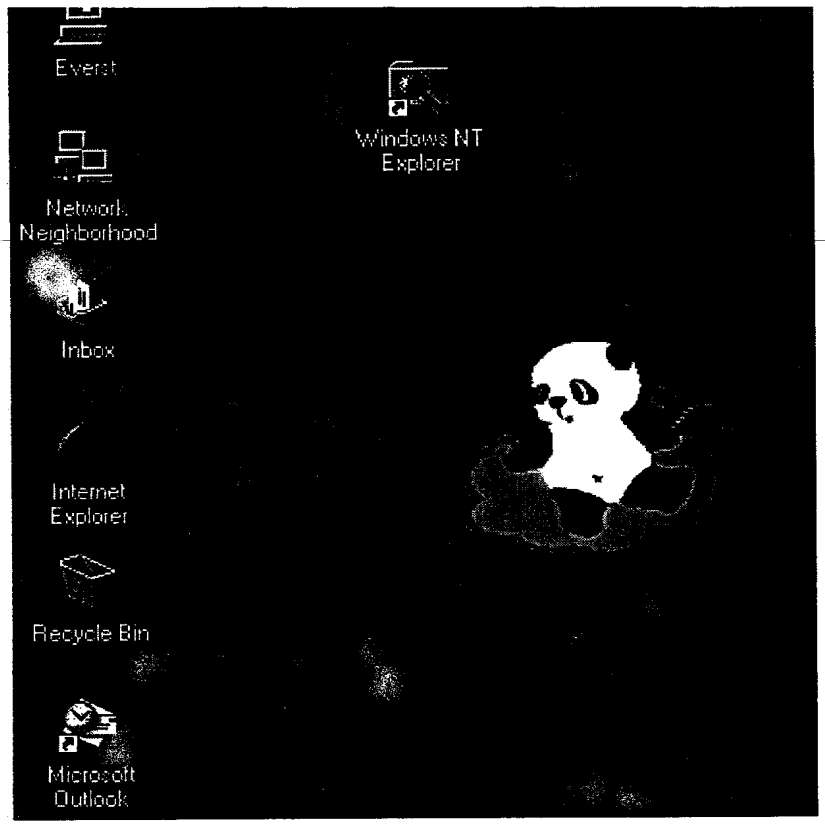
```
//links to other objects:
CQTCharController* pCharController;
CQTDisplayController* pQTDisplayController;

CQTCharacter*     pCharacter;
CQTHostSystem* system_info;

//owner info:
//int       userID;
//LPSTR  userName;


//internal data:
int clock_interval;
DWORD brainState;


time_t lastUserActionTime; //used to determine what to do after certain idle time
                                                                    //for comic pet

//server command handling:
int          serverCommand;  //invocation method of server script
CString serverScript;   //script text, loadedfrom server


//imagemap handling:
bool ImageMapBufferFull();
int  ImageMapBufferSet(CString objectName, CString frameName);

CString imageMapEventObjectName;    //these are the imagemap buffer;
CString imageMapEventFrameName;


CQTBrainImageMap* pImageMap;   //this is the imagemap object;
CString imageMapName;       //to speed up things, we may prefer to first load the script into
current_exec_space
                                                    //into current_exec_space, start executing, and put the
corresponding imageMap name into
                                                                    //imageMapName, and send self
(Brain) a message to load the map while executing the script.


//current script engine execution space:
CQTScriptEngine* pCurrentExecSpace;
int pausePC;  //when we pause execution of script, we store the ProgramPointer here.
int allowInterrupt; //if this is set to 1, interrupt of script execution is allowed  (when doing
imagemapping)

//queued script engine space:
```

```
        //global script-engine-variable list:
        //global script-engine-object list:
        int petType;  //options: comic, game, messenger (customize), assistant (chat/scavenger hunt), living
(eat/sleep/grow)

        CQTTicGame* pGameObject;
        CQTKnockoutScoreboard* pKnockoutScoreboard;

        // Messenger Pet Interface:

        CQTCommandMessenger* pMessengerWindow;
        CQTCommandChat* pChatWindow;
        CQTCommandEmail* pEmailWindow;


};

#endif // !defined(AFX_QTBRAIN_H__83213185_941C_11D2_9641_00105A9E5870__INCLUDED_)
```

APPENDIX B


JJ Mountain
Submit


Everest
Windows NT Explorer
Network Neighborhood
Inbox
Internet Explorer
Am I cute? you don't have to answer, never mind ...
Recycle Bir
Microsoft Outlook
Publish
Setup for Microsoft I...


Everst
Windows NT Explorer
Network Neighborhood
Inbox
Internet Explorer
Recycle Bin
Microsoft Outlook

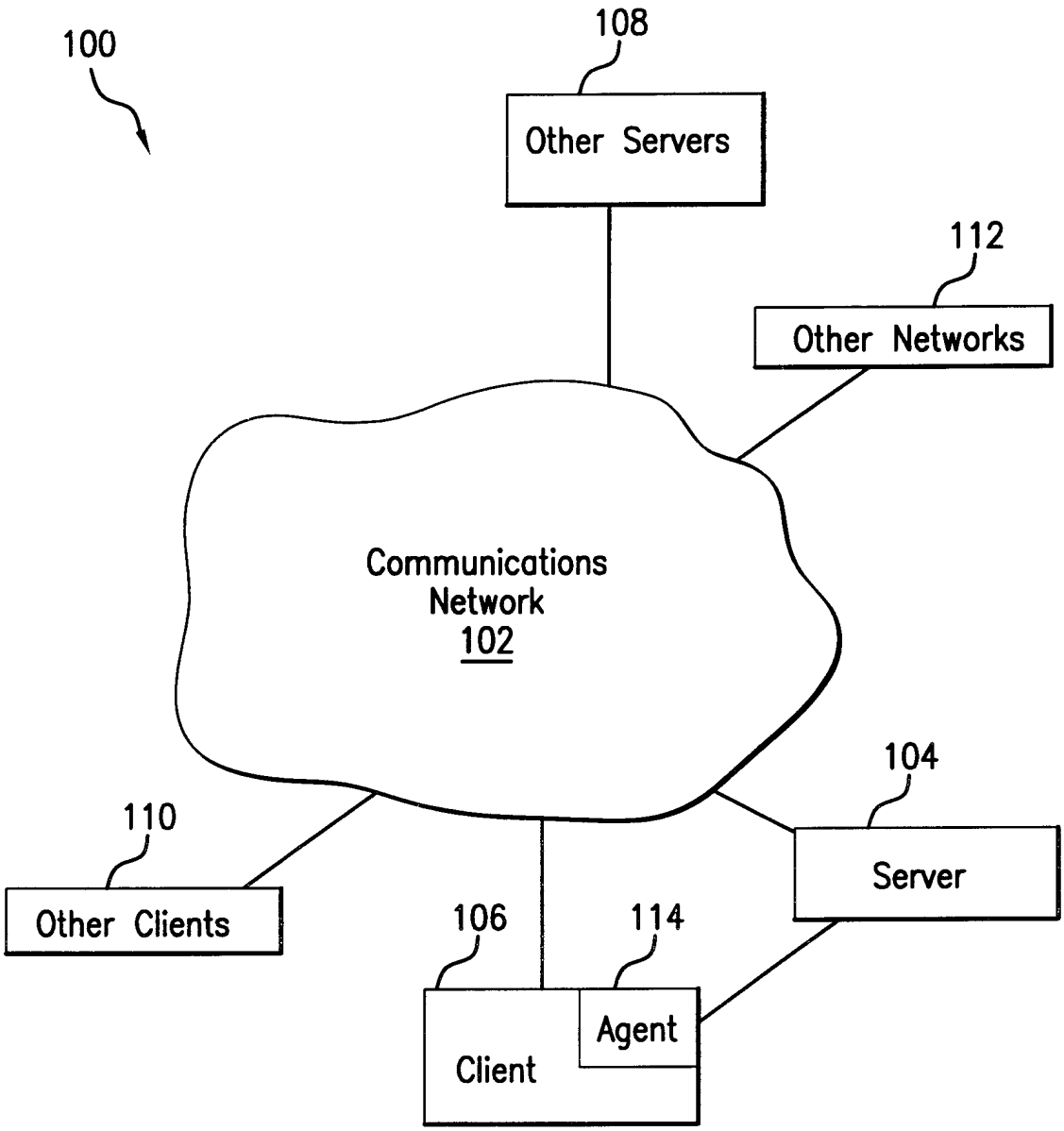

100
108
Other Servers
112
Other Networks
Communications
Network
102
104
Server
110
Other Clients
106
114
Agent
Client

FIG.1

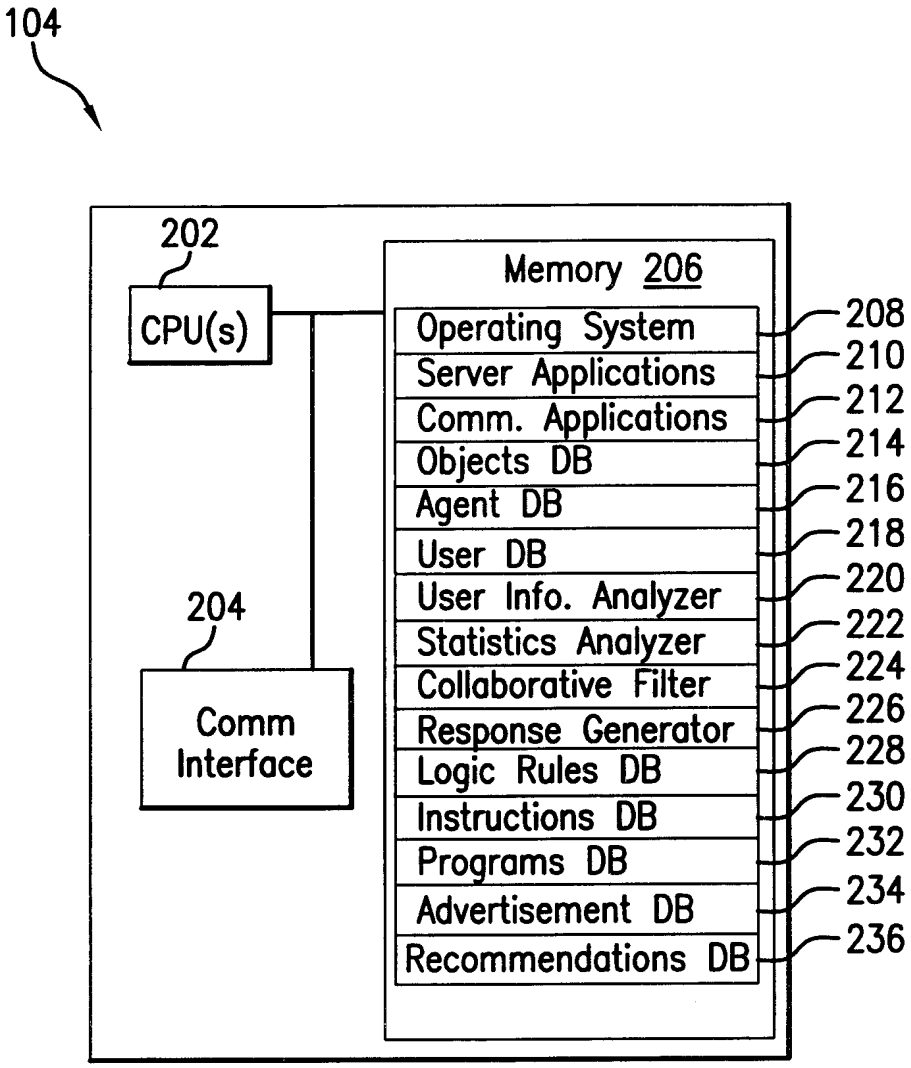

104
202
CPU(s)
204
Comm Interface
Memory 206
Operating System
Server Applications
Comm. Applications
Objects DB
Agent DB
User DB
User Info. Analyzer
Statistics Analyzer
Collaborative Filter
Response Generator
Logic Rules DB
Instructions DB
Programs DB
Advertisement DB
Recommendations DB
208
210
212
214
216
218
220
222
224
226
228
230
232
234
236

FIG.2

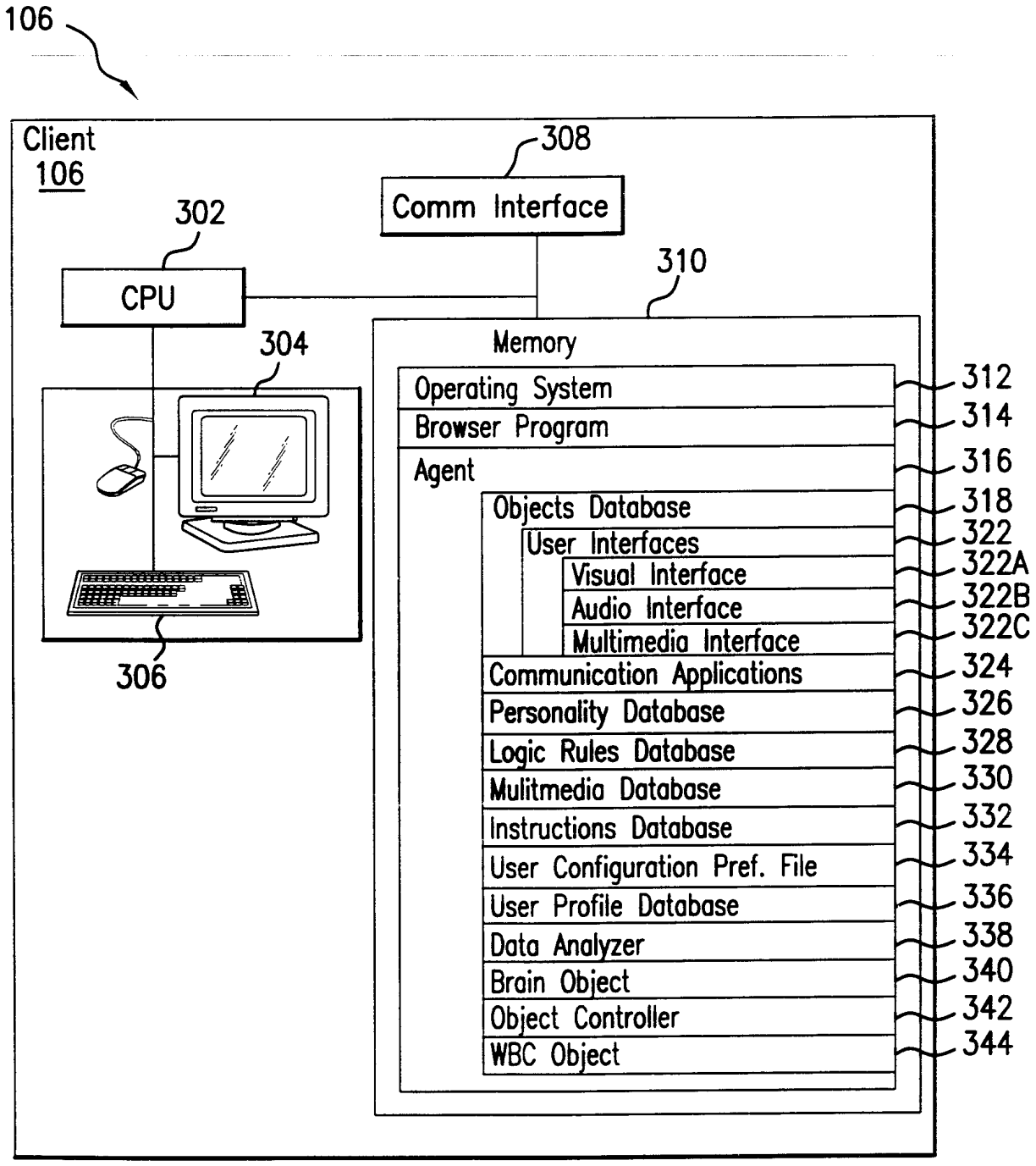

106
Client
106
308
Comm Interface
302
CPU
310
304
Memory
Operating System
Browser Program
Agent
Objects Database
User Interfaces
Visual Interface
Audio Interface
Multimedia Interface
Communication Applications
Personality Database
Logic Rules Database
Mulitmedia Database
Instructions Database
User Configuration Pref. File
User Profile Database
Data Analyzer
Brain Object
Object Controller
WBC Object
306
312
314
316
318
322
322A
322B
322C
324
326
328
330
332
334
336
338
340
342
344

FIG.3A

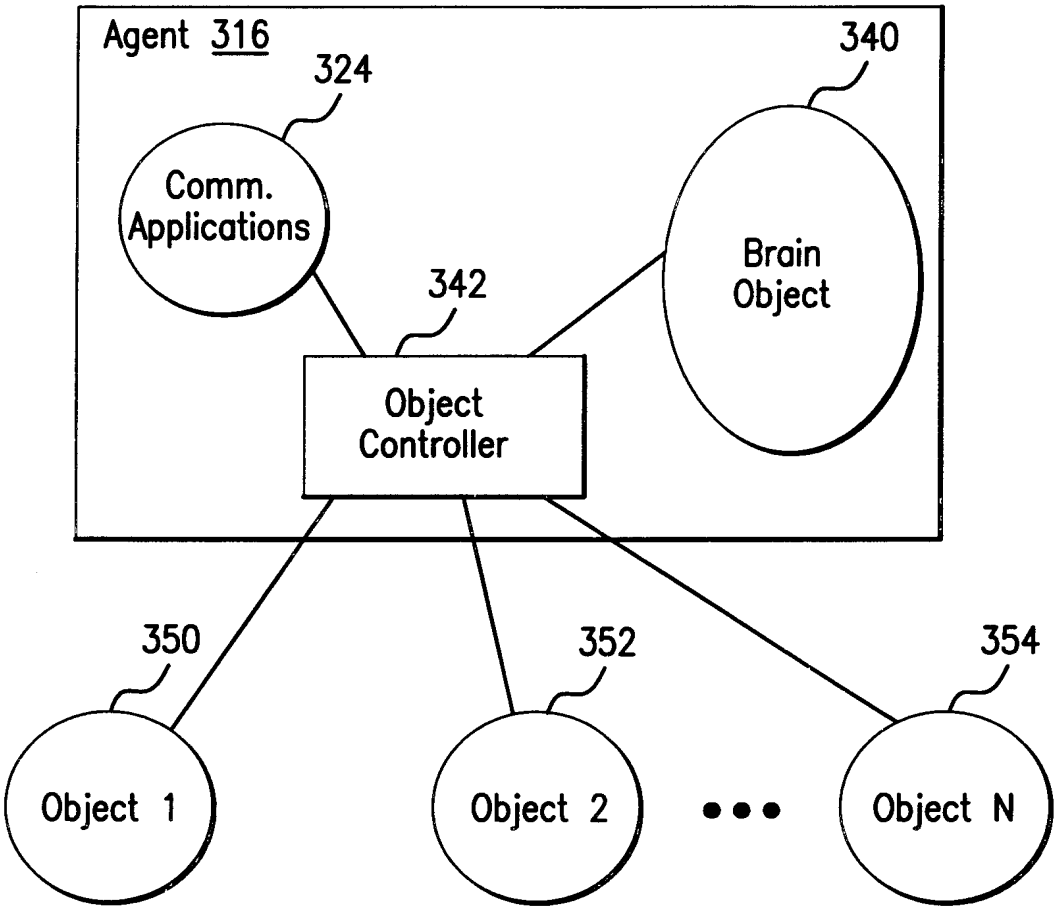

Agent 316
324
340
Comm. Applications
Brain Object
342
Object Controller
350
352
354
Object 1
Object 2
Object N

FIG.3B

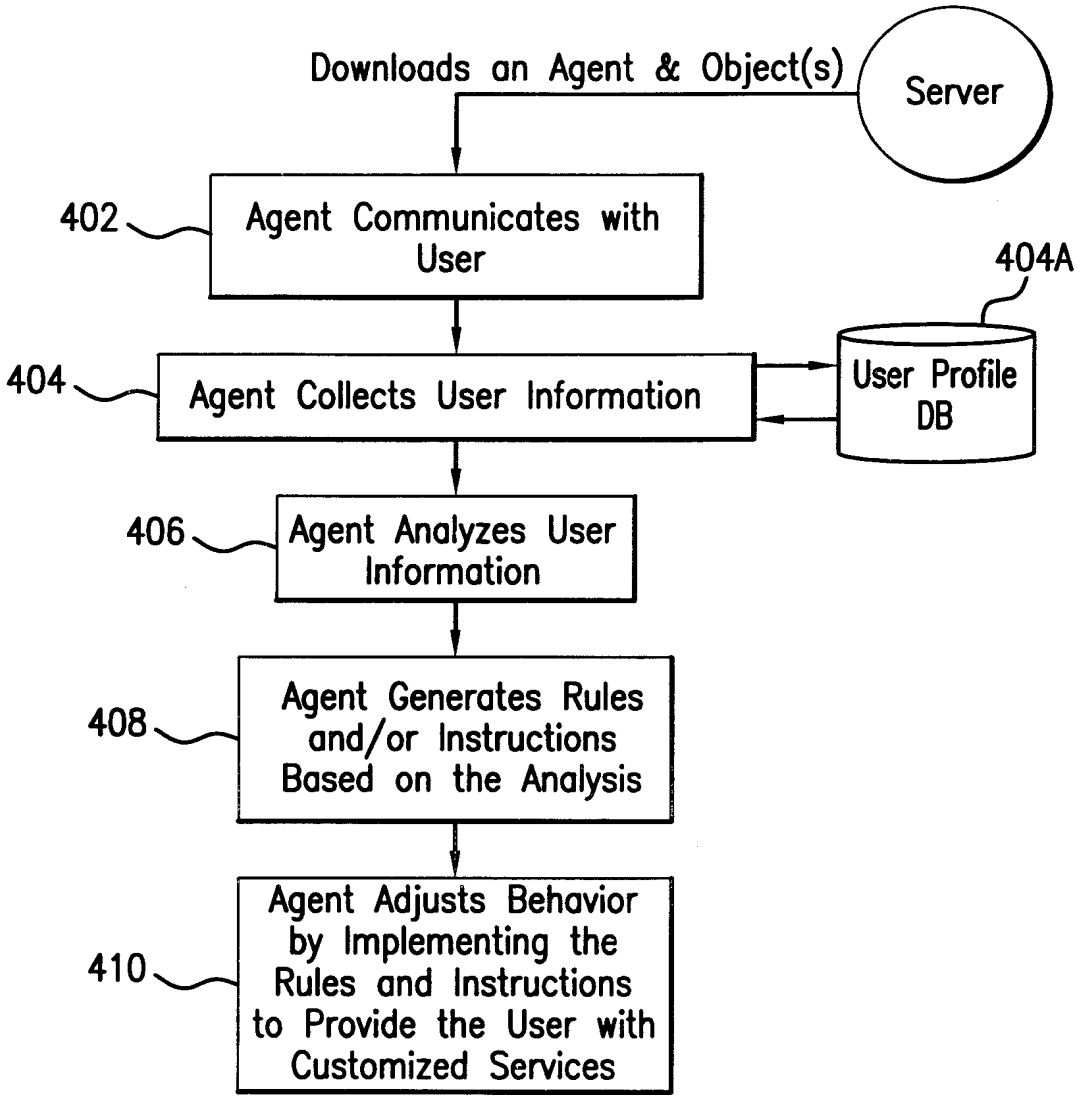

Downloads an Agent & Object(s)
Server
402
Agent Communicates with User
404A
404
Agent Collects User Information
User Profile DB
406
Agent Analyzes User Information
408
Agent Generates Rules and/or Instructions Based on the Analysis
410
Agent Adjusts Behavior by Implementing the Rules and Instructions to Provide the User with Customized Services

FIG.4

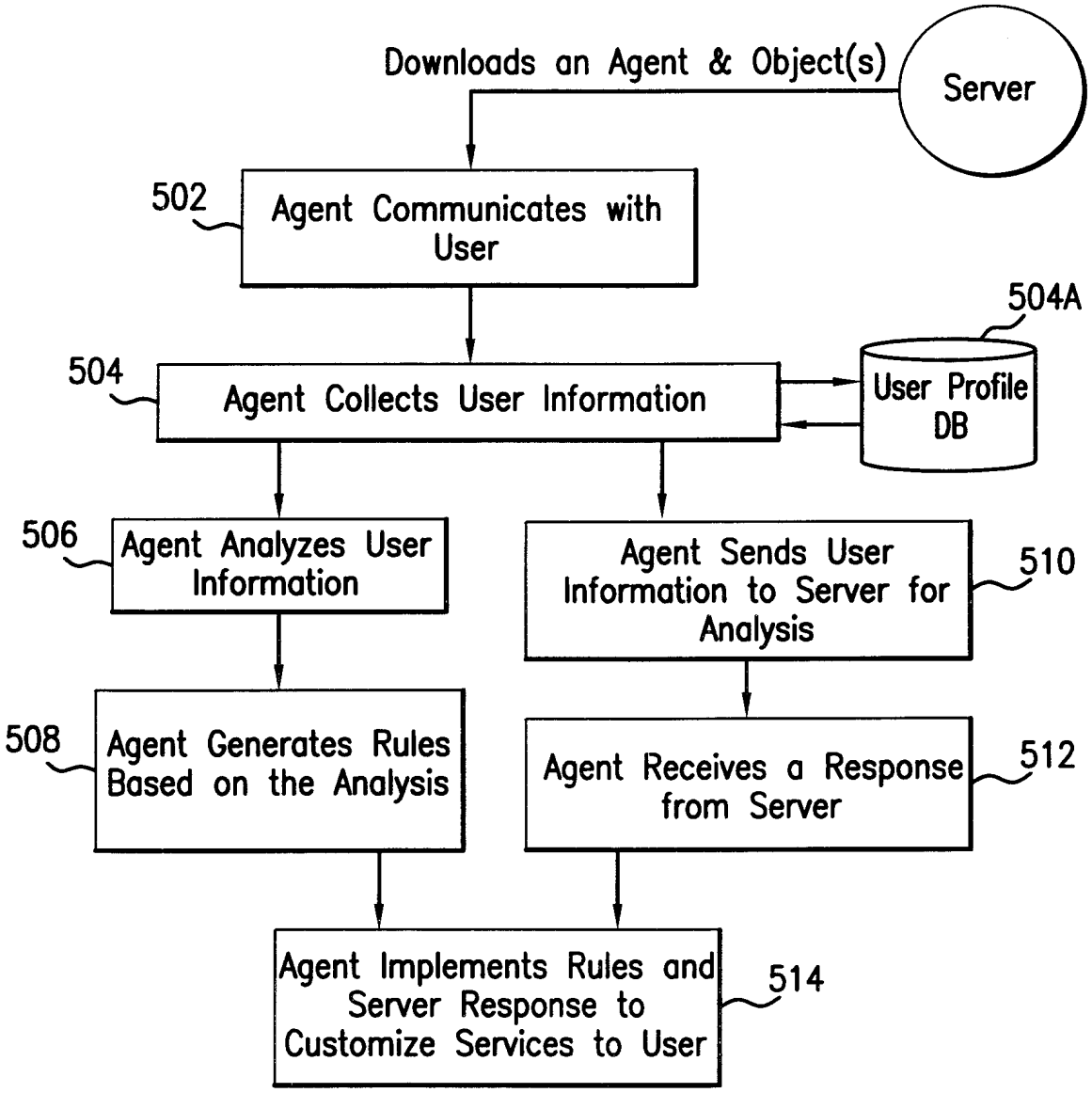

Downloads an Agent & Object(s)
Server
502
Agent Communicates with User
504A
504
Agent Collects User Information
User Profile DB
506
Agent Analyzes User Information
Agent Sends User Information to Server for Analysis
510
508
Agent Generates Rules Based on the Analysis
Agent Receives a Response from Server
512
Agent Implements Rules and Server Response to Customize Services to User
514

FIG.5

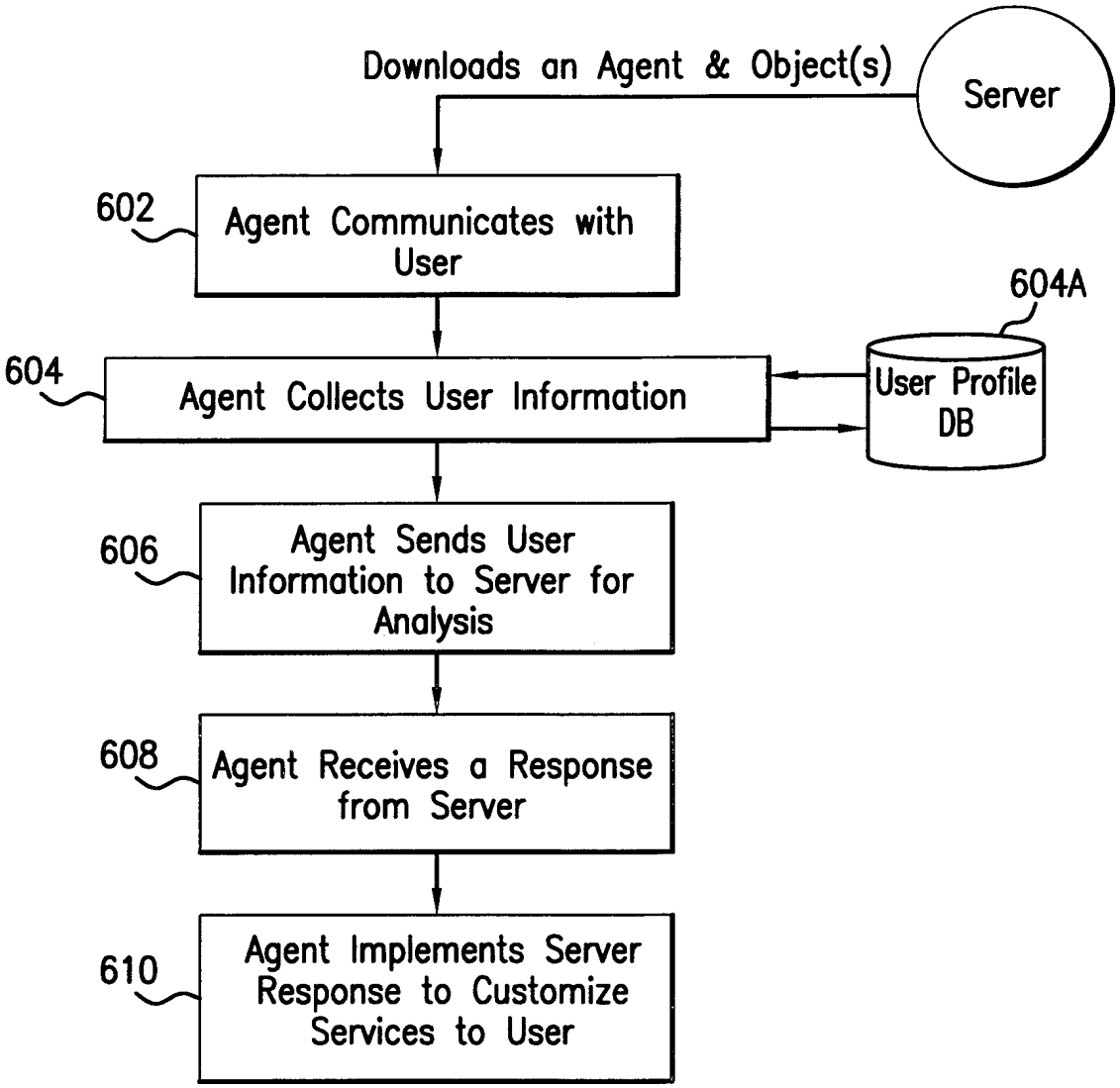

Downloads an Agent & Object(s)
Server
602
Agent Communicates with User
604A
604
Agent Collects User Information
User Profile DB
606
Agent Sends User Information to Server for Analysis
608
Agent Receives a Response from Server
610
Agent Implements Server Response to Customize Services to User

FIG.6

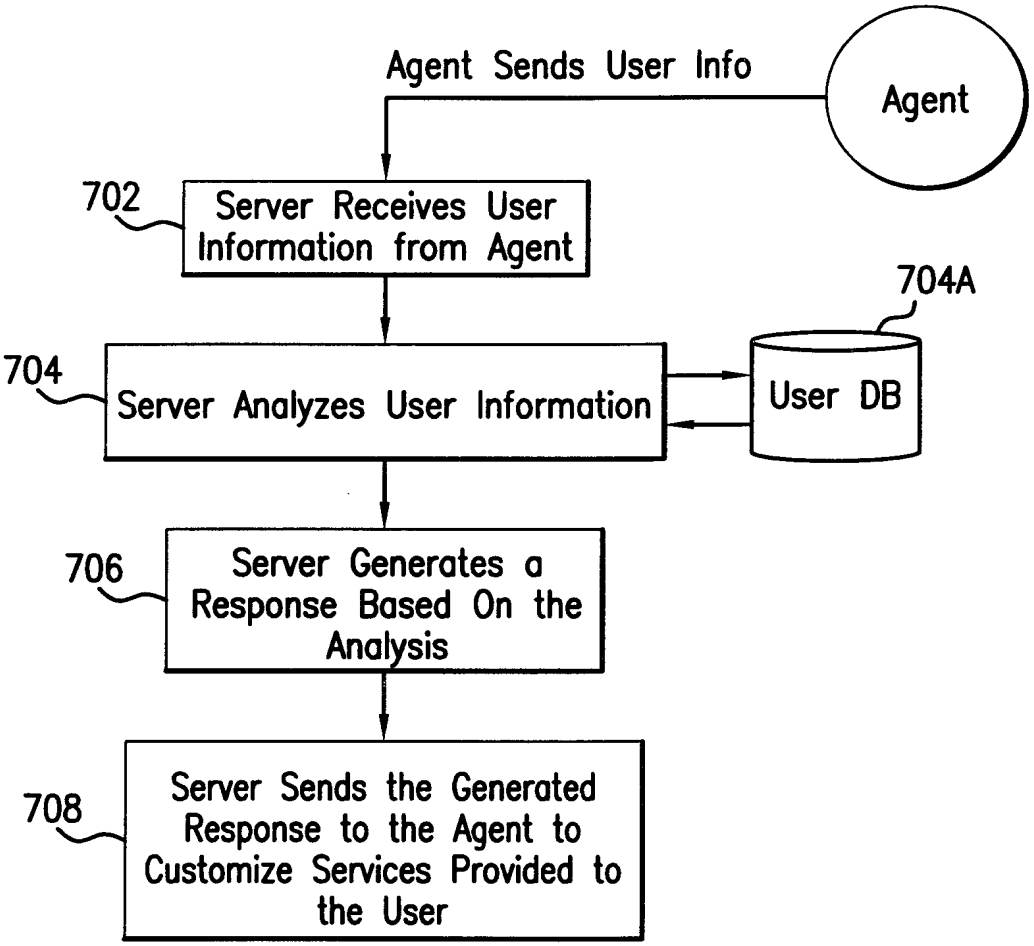

Agent Sends User Info
Agent
702
Server Receives User Information from Agent
704A
704
Server Analyzes User Information
User DB
706
Server Generates a Response Based On the Analysis
708
Server Sends the Generated Response to the Agent to Customize Services Provided to the User

FIG.7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/30580

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :Please See Extra Sheet.
US CL : 709/224; 705/10
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 709/224; 705/10

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X ----- Y	US 5,796,952 A (DAVIS et al) 18 August 1998, col. 4, line 3-col. 5, line 3, col. 6, lines 14-34, col. 8, lines 6-52, col. 9, lines 11-45.	1 --- 2-53
Y	US 5,848,396 A (GERACE) 08 December 1998, col. 2. line 3-col. 3. line 10, col. 4. lines 1-47, col. 5. lines 8-40, col. 6. line 22- col. 7. line 3, col. 11, lines 24-42, col. 13, lines 20-26.	2-53

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 25 FEBRUARY 2000	Date of mailing of the international search report 17 MAR 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer LE HIEN LUU Telephone No. (703) 305-9650

Form PCT/ISA/210 (second sheet)(July 1992)★

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/30580

A. CLASSIFICATION OF SUBJECT MATTER:
IPC (6):

G06F 15/16, 15/173, 17/60